A function with a return value produces a value that it returns to the function that called it. In other words, if the function returns the square root of 9.0 \((\sqrt{9.0})\), then the function call has the value 3.0. Such a function is declared as having the same type as the value it returns. Here is the general form:

\[
\text{typeName functionName}(\text{parameterList})
\{
\hspace{1em} \text{statements}
\hspace{1em} \text{return value;} \hspace{0.5em} // \text{value is of type typeName}
\}
\]

Functions with return values require that you use a return statement so that the value is returned to the calling function. The value itself can be a constant, a variable, or a more general expression. The only requirement is that the expression reduce to a value that has, or is convertible to, the \text{typeName} type. (If the declared return type is, say, \text{double}, and the function returns an \text{int} expression, the \text{int} value is typecast to type \text{double}.) The function then returns the final value to the function that called it. C++ does place a restriction on what types you can use for a return value: The return value cannot be an array. Everything else is possible— integers, floating-point numbers, pointers, even structures and objects! (Interestingly, even though a C++ function can't return an array directly, it can return an array that's part of a structure or object.)

As a programmer, you don't need to know how a function returns a value, but knowing the method might clarify the concept for you. (Also, it gives you something to talk about with your friends and family.) Typically, a function returns a value by copying the return value to a specified CPU register or memory location. Then, the calling program examines that location. Both the returning function and the calling function have to agree on the type of data at that location. The function prototype tells the calling program what to expect, and the function definition tells the called program what to return (see Figure 7.1). Providing the same information in the prototype as in the definition might seem like extra work, but it does make good sense. Certainly, if you want a courier to pick up something from your desk at the office, you enhance the odds of the task being done right if you provide a description of what you want both to the courier and to someone at the office.

Figure 7.1. A typical return value mechanism.